

性能評価モデル生成装置および性能評価モデル生成方法

APPARATUS AND METHOD FOR PRODUCING PERFORMANCE EVALUATION MODEL

発明の背景

発明の分野

- 5 本発明は、本発明は性能評価モデル生成装置および性能評価モデル生成方法に関し、特にUML (Unified Modeling Language)モデルから性能評価モデルを生成する性能評価モデル生成装置および性能評価モデル生成方法に関する。

背景技術

- 10 製品サイクルの短期化のため、サービス分析から実現までの時間的短縮が望まれている。また、組み込み機器分野においては、情報端末の普及によって、より多機能かつ高機能なシステムが求められ、また、システムL S I (Large Scaled Integration)の登場によって、ハードウェアとソフトウェアとをより密にシステム設計する必要性が生じている。
- 15 このような状況の下で、システム構成を決めて実装した後に評価を行っているようでは、時間的な短縮は望めない。そこで、システム構成を決定するに当たって、実装することなく性能などの見積もりを行う必要性が生じてくる。このニーズに対して、待ち行列理論を使った性能評価ツールが使われている。このような性能評価ツールでは、システムに入力されるジョブを待ち行列を使ってプロセッサに割り当てるという性能評価モデルを作成し、処理速度などのパラメータを与えることで性能評価を行っている。この性能評価モデルを使った性能評価は、性能評価モデルを作成するコストが高いため、強力な性能評価ができるにもかかわらず、広く使われるに至っていない。
- 20 特開平7-84831号公報では、このような性能評価の支援手法として、過去に性能評価したモデル（性能評価モデル）をデータベースに保持し、以降の設計モデルの作成に役立てるようなツールが提案されている。このツールでは、性能評価をある一定期間続けることでツールに依存することなく性能評価モデルが蓄積されて、設計モデルの作成が容易になるというメリットが得られる。

- 25 特開平7-84831号公報では、このような性能評価の支援手法として、過去に性能評価したモデル（性能評価モデル）をデータベースに保持し、以降の設計モデルの作成に役立てるようなツールが提案されている。このツールでは、性能評価をある一定期間続けることでツールに依存することなく性能評価モデルが蓄積されて、設計モデルの作成が容易になるというメリットが得られる。

しかし、上述した従来のツールによっても、システムをゼロから開発する場合で、性能評価モデルに類似のものが無いときには、性能評価モデルをゼロから作成する必要があるため、性能評価モデルを作成するコストが高いという問題があった。

5

発明の要旨

本発明の目的は、モデル作成のための事実上の標準言語であるUMLで記述されたUMLモデルから性能評価モデルを自動的に生成する性能評価モデル生成装置を提供することにある。

- 10 また、本発明の他の目的は、UMLで記述されたUMLモデルから性能評価モデルを自動的に生成する性能評価モデル生成方法を提供することにある。

本発明の第1の態様の性能評価モデル生成装置は、変換規則を格納する変換規則格納部と、UMLモデルを入力して解析するUMLモデル解析部と、前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。

- 15 また、本発明の第2の態様の性能評価モデル生成装置は、利用者からの変換規則を入力する変換規則編集部と、前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、UMLモデルを入力して解析するUMLモデル解析部と、前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。

- 20 本発明の性能評価モデル生成方法は、UMLモデルの拡張記法を使い、ハードウェアリソースに割り付ける表記を定め、その表記に従ったUMLモデルから性能評価モデルへの変換規則を決め、この変換規則によってUMLモデルから系統的に性能評価モデルを作成する。

- 25 本発明の他の性能評価モデル生成方法は、シーケンス図の最初のメッセージをカレントメッセージにセットする工程と、シーケンス図のカレントメッセージに着目する工程と、送信先オブジェクトを定義するクラスのタイプおよび属性を調

- 査する工程と、送信元オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに基づいて変換規則を検索する工程と、検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、カレントメッセージの次のメッセージがあるかどうかを判定する工程と、次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメッセージに着目する工程に戻す工程とを含む。特に、前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなる。
- 15 本発明の記録媒体は、コンピュータを、変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。
- 20 また、本発明の記録媒体は、コンピュータを、利用者からの変換規則を入力する変換規則編集部、前記変換規則編集部により入力された変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。
- 25

本発明の第1の効果は、システムの分析および設計で広く使われているUMLモデルを元に、性能評価モデルを作成することにより、性能評価モデルの作成が容易になり、性能評価モデルの作成にかかる時間の削減、つまりコストの削減が

できることである。

その理由は、UMLモデルで純粋に機能モデルを作成する部分に関しては、これまでのシステム設計の手法をほぼそのまま流用でき、UMLモデルを元にしてハードウェアリソースとの対応を考えるだけで性能評価モデルが生成できるためである。

第2の効果は、システムのUMLモデルに誤りが無ければ、性能評価モデルの作成時点で起こりうる性能評価モデルの作成誤りをなくすることが可能となることである。

その理由は、UMLモデルに誤りがなく、そのUMLモデルから系統的に性能評価モデルを作成するため、性能評価モデルの作成時に起こりうる単純な誤り、勘違いなどを防ぐことが可能となるからである。

第3の効果は、性能評価ツールの違いを越えて、既存のモデルや他の人が作成したモデルを広く参照することが容易になることである。

その理由は、機能モデルとしてUMLという統一モデリング言語を使うことにより、異なる性能評価ツール間でも同一のUMLモデルを使ってシステムを表現できるためである。性能評価モデルは、一般的に、独自のフォーマットでモデルを記述する必要がある、このモデルは一般的に各ツールで異なっている。本発明によれば、UMLモデルという統一表記法でシステムの機能モデルを入力し、性能評価をするため、システムの機能モデルの入力も一般的なツールが使える、性能評価ツールの違いを越えて、既存資産の流用も容易となるからである。

図面の簡単な説明

図1は、本発明の第1の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。

図2は、第1の実施例に係る性能評価モデル生成方法の概要を説明する図である。

図3は、図2中のシーケンス図の内容を例示する図である。

図4は、図2中のクラス図の内容を例示する図である。

図5は、図2中の性能評価サブモデルの内容を例示する。

図6は、第1の実施例に係る性能評価モデル生成方法の手順を示すフローチャートである。

図7は、リソースクラスに対応するノード群を例示する図である。

- 5 図8は、本発明の第2の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。

望ましい実施態様

以下、本発明の実施例について図面を参照して詳細に説明する。

10

第1の実施例

- 図1は、本発明の第1の実施例に係る性能評価モデル生成装置の構成を示すブロック図である。本実施例に係る性能評価モデル生成装置は、UMLモデル5を入力して解析するUMLモデル解析部1と、UMLモデル解析部1による解析結果を変換規則格納部4に格納された変換規則に従って変換して性能評価モデル6を生成する変換処理部2と、利用者からの変換規則を入力する変換規則編集部3と、変換規則編集部3により入力された変換規則を格納する変換規則格納部4とから構成されている。なお、図1中、符号7はUMLモデル5を生成するUMLモデル入力ツール、8は性能評価モデル6を性能評価する性能評価ツールをそれぞれ示す。
- 15 20

- 図2は、システムの動作を記述したUMLモデル5から性能評価モデル6を系統的に生成する性能評価モデル生成方法を説明する図である。UMLモデル5は、全てUMLという統一モデリング言語の規格に従った記法で記述する。システムの動作は、動的な振る舞いを表すシーケンス図51と、システムの静的な特徴を表すクラス図52とで表現される。第1の実施例では、UMLモデル5の記述単位として、UMLモデル5をパッケージ50という単位に分割する。つまり、UMLモデル5は1つ以上のパッケージ50から構成され、パッケージ50毎にシーケンス図51およびクラス図52を用意する。また、1つ
- 25

のパッケージ50内のシーケンス図51およびクラス図52から生成される性能評価モデル6の単位を、性能評価サブモデル60と呼ぶ。つまり、性能評価モデル6は、1つ以上の性能評価サブモデル60から構成されている。

図3を参照すると、シーケンス図51は、システムの動的な関係をモデル化した図である。シーケンス図51の構成要素の中で、性能評価サブモデル60の作成に関係するものには、オブジェクトOb01～Ob06と、メッセージM01～M07とがある。オブジェクトOb01～Ob06は、機能単位に登場するクラスのインスタンスである。具体的には、後述する図4のクラス図52中に対応するクラスC01～C05のインスタンスである。メッセージM01～M07は、クラスC01～C05の実際の動作手順を時系列に沿って配置したものであり、上の方が相対的に早い時間を表している。破線矢印で示されたものは、手続き呼び出しの戻りを示している。戻りの破線矢印は省略してもよい。

図4は、クラス図52およびクラス図52に対して性能評価サブモデル60を作成するために行われる拡張表記を説明する図である。クラス図52は、システムを構成するクラスの静的な関連を示した図である。クラスは、モデリングに使用されるモデル要素であり、システムを構成する通常のオブジェクト指向設計で使われるクラス（以下、システムクラスという）と、ハードウェアリソースを表すクラス（以下、リソースクラスという）とに大きく区別され、それぞれさらに細かく分類される。この分類については、以下の[クラスの分類]で詳しく説明する。図4においては、クラスC01～C05がシステムクラスである。性能評価のためにクラス図52に付加する拡張表記は、1つが、処理に必要なハードウェアリソースを表すリソースクラスC11, C12, C13として記述する。もう1つは、これらのハードウェアリソースへの関連記述であり、これをリソース関連線L01, L02, L03と名付ける。また、関連自体をリソース関連と名付け、システムクラスC02, C03, C04をリソース関連線L01, L02, L03を使ってリソースクラスC11, C12, C13と結び付けることで、リソースクラスC11, C12, C13のハード

ウェアリソースにシステムクラスC02, C03, C04を割り付けるという意味づけをする。このリソース関連により、性能評価サブモデル60の生成を可能としている。変換規則格納部4中の変換規則は、以下に示す[クラスの分類]と、[性能評価モデルの変換規則]とで構成されている。

5

[クラスの分類]

◎システムクラス

- ・タイプ<<Active>>のクラス

通常のクラス。属性およびメソッドを持ち、自立的に動作するクラス。

- 10 ・タイプ<<Data>>のクラス

属性が中心のクラス。メソッドは、所有する属性の読み書きのみに限定されている。

- ・タイプ<<Interface>>のクラス

- 15 タイプ<<Interface>>のクラスは、パッケージ50内に唯一のクラスであり、外部パッケージからカレントパッケージへの全てのメッセージを受け取り、カレントパッケージ内の各クラスにメッセージを送出する。

- ・タイプ<<Actor>>のクラス

- 20 タイプ<<Actor>>のクラスは、システムの外部に存在し、外部からシステム内に何らかの働きかけを行うクラス的一种である。アクターがシステム内部にも記憶域を持つ場合、クラス図52にタイプ<<Actor>>のクラスとタイプ<<Storage>>のクラスとのクラス関連を記述する関係上、アクターの記述が必要となる(後述する[変換規則]のa.アクターを参照)。これ以外の場合、クラス図52にアクターの記述がなくてもよい。

◎リソースクラス

- 25 リソースクラスは、ハードウェアリソースを表すクラスである。リソースクラスには、性能評価サブモデル60に対応するノード群がある。ハードウェアリソースの占有時間など、変換したノードに設定すべきパラメータをクラスの属性に記述し、そのパラメータを使って性能評価サブモデル60のノードに変

換する。性能評価の対象にする必要がない場合、評価対象外と記し、性能評価サブモデル60のノードは生成しない。リソースクラスが対応する性能評価サブモデル60のノード群の例を、図7に示す。

・タイプ<<Storage>>のクラス

- 5 タイプ<<Storage>>のクラスは、ハードウェアリソースの中で記憶域を定義するクラスである。記憶域の種類や性質に応じて、タイプ<<Storage>>でさまざまなクラスを定義する。

・タイプ<<Processing>>のクラス

- 10 タイプ<<Processing>>のクラスは、ハードウェアリソースの中で実際の処理をするハードウェアを定義するクラスである。ソフトウェア処理の場合のCPU (Central Processing Unit) も、このクラスで定義する。

[性能評価モデルへの変換規則]

- 15 メッセージの送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに応じて、以下の変換規則を適用する。クラスのタイプを追加することで、この変換規則を追加することが可能となる。以下、「送信元オブジェクトを定義するクラスのタイプ → 送信先オブジェクトを定義するクラスのタイプ」という形式で変換規則を記述する。

- 20 a. 「タイプ<<Actor>> → 任意のクラスタイプ」

タイプ<<Actor>>のクラスが最初のメッセージを送信する送信元オブジェクトになっている場合、データの発生源であるソースノードに変換する。

b. 「外部パッケージの任意のクラスタイプ → カレントパッケージのタイプ <<Interface>>」

- 25 カレントパッケージのタイプ<<Interface>>のクラスに対するメッセージは、データの入り口であるエンターノードに変換する。

c. 「カレントパッケージのタイプ<<Interface>> → 外部パッケージの任意のクラスタイプ」

カレントパッケージのタイプ<<Interface>>のクラスから外部パッケージへのメッセージは、データの出口であるリターンノードに変換する。

d. 「任意のクラスタイプ → タイプ<<Processing>>」

5 タイプ<<Processing>>のクラスが定義するリソースノードを配置する。

e. 「任意のクラスタイプ → タイプ<<Storage>>」

タイプ<<Storage>>のクラスが定義するリソースノードを配置する。

f. 「任意のクラスタイプ → 外部パッケージのタイプ<<Interface>>」

10 外部パッケージのタイプ<<Interface>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価サブモデル60に変換する。これに続くメッセージが、カレントメッセージと逆方向になっている場合、これに続くメッセージを無視する。

15 g. 破線矢印

破線矢印のメッセージは、それ自体省略しても可能なものである。よって、このメッセージは、変換の際に無視することができる。

h. 終了処理

最後のメッセージでシンクノードまたはリターンノードが配置されない場合、
20 このどちらかのノードを配置し、性能評価サブモデル60を完結させる。どちらのノードを配置するかは、変換した性能評価サブモデル60の最初のノードに着目し、シンクノードの場合はソースノード、エンターノードの場合はリターンノードを配置する。

図5を参照すると、性能評価サブモデル60は、システム構成をノードで表現する。ノードは、大別すると、システムの処理フロー自体を表すノードと、
25 処理フローで利用されるハードウェアリソースを表すノードとに区別できる。図5中の性能評価サブモデル60では、前者が下側に、後者が上側に記述されている。システムの処理フローを表現するノードは、そのフローをノード間の

結線で表現している。このようなノードには、ハードウェアを消費する「処理」を表すサービスノード、データの入力を表すソースノード、処理の終了を表すシンクノードなどがある。一方、ハードウェアリソースを表すノードとして、ハードウェアに相当するリソースノードがある。リソースノードは、サービスノードでの処理によって消費される。この消費時間、優先順位などを動的パラメータ 6 1 として設定する。さらに、入力されるデータのパラメータを加えて、性能評価サブモデル 6 0 が生成できる。

図 5 は、性能評価サブモデル 6 0 の一例を記述する図である。なお、ここでは、性能評価ツール 8 として、待ち行列理論を使った S E S / W o r k b e n c h (S E S 社の登録商標) を想定している。この性能評価サブモデル 6 0 には、処理する側のノードとして、上部に、C P U を表すサービスノード N 0 1 が記述されている。一方、処理フローを表すノードとして、下部に、C P U を消費するサービスノード N 0 3 と、C P U とは無関係に時間を消費するサービスノード N 0 5 と、システムに入力されるデータに関連するソースノード N 0 2 およびシンクノード N 0 8 と、その他のノードとして、性能評価サブモデル 6 0 への参照を表すサブモデルノード N 0 7, リソース管理のアロケートノード N 0 4, リリースノード N 0 6 と、そして、その間の結線 3 1 1 ~ 3 1 6 が記述されている。最後に、上部と下部とを結び付ける記述として、動的パラメータ 6 1 が設定されている。変換では、リソース管理のアロケートノード N 0 4, メモリアクセスを表すサービスノード N 0 5, およびリリースノード N 0 6 という一連のノードを排他アクセス M e m o r y のモデルとしてあらかじめ用意しておき、これらをセットにして変換規則を適用する。図 5 にあるノード以外に、他の性能評価サブモデル 6 0 からデータの流れを引き継ぐエンターノードおよびリターンノードがある。

25

[性能評価サブモデルへの変換手順]

図 6 は、性能評価サブモデル 6 0 への変換手順を表すフローチャートである。性能評価サブモデル 6 0 は、あるパッケージ 5 0 に着目し、そのパッケージ 5

0のシーケンス図5 1およびクラス図5 2から変換して生成される。この着目しているパッケージ5 0をカレントパッケージと呼び、それ以外のパッケージ5 0を外部パッケージと呼ぶ（図4参照）。

- 図6を参照すると、変換処理部2の処理は、カレントメッセージセットステップS 1 0 1と、カレントメッセージ着目ステップS 1 0 2と、送信元オブジェクトのクラスタイプおよび属性調査ステップS 1 0 3と、送信先オブジェクトのクラスタイプおよび属性調査ステップS 1 0 4と、変換規則検索ステップS 1 0 5と、ノード生成・連結ステップS 1 0 6と、次メッセージ有無判定ステップS 1 0 7と、オブジェクト着目ステップS 1 0 8と、クラス着目ステップS 1 0 9と、リソース関連探索ステップS 1 1 0と、リソース関連有無判定ステップS 1 1 1と、リソース関連先クラスタイプおよび属性保存ステップS 1 1 2と、自クラスタイプおよび属性保存ステップS 1 1 3とからなる。

次に、このように構成された第1の実施例に係る性能評価モデル生成装置の動作について、第1の実施例に係る性能評価モデル生成方法とともに説明する。

- ここでは、図3のシーケンス図5 1および図4のクラス図5 2からメッセージを時系列に沿って図5の性能評価サブモデル6 0に変換する例について説明する。利用者は、変換規則編集部3を使用して、あらかじめ変換規則を編集して変換規則格納部4に格納しておく。ここでは、既述した変換規則a. ～h. が格納されたものとする。

- 一方、UMLモデル解析部1は、UMLモデル入力ツール7を使って生成されたUMLモデル5を入力して解析する。詳しくは、性能評価サブモデル6 0に変換するために不必要な情報を削除し、必要な情報が揃っているかどうかをチェックし、必要な情報が揃っていたならば変換処理部2に解析結果を渡す。

- 変換処理部2は、UMLモデル解析部1から渡された解析結果中の、シーケンス図5 1の最初のメッセージM 0 1をカレントメッセージにセットする（ステップS 1 0 1）。

次に、変換処理部2は、シーケンス図5 1のカレントメッセージM 0 1に着目する（ステップS 1 0 2）。

詳しくは、変換処理部 2 は、まず、送信元オブジェクト O b 0 1 に着目し (ス

- 次に、変換処理部 2 は、カレントメッセージ M 0 1 の送信先オブジェクト O b 0 2 を定義するクラス C 0 1 のタイプおよび属性を調査する（ステップ S 1 0 4）。

- 続いて、変換処理部 2 は、送信元オブジェクト O b 0 1 を定義するクラスのタイプ<<A c t o r>>および送信先オブジェクト O b 0 2 を定義するシステムクラス C 0 1 のタイプ<<I n t e r f a c e>>に基づいて、変換規則格納部 4

- 続いて、変換処理部 2 は、検索された変換規則 a.「アクターが最初のメッセージを送信するオブジェクトになっている場合、データの発生源であるソースノードに変換する。」に基づいて、ソースノード N02 に変換して性能評価サブ

モデル60に配置する（ステップS106）。

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し（ステップS107）、次のメッセージM02があるので、ステップS102に制御を戻す。変換処理部2は、次のメッセージM02をカレントメッセージに
5 セットして着目する（ステップS102）。

次に、変換処理部2は、メッセージM02の送信元オブジェクトOb02を定義するシステムクラスのタイプおよび属性を調査する（ステップS103）。詳しくは、変換処理部2は、まず、送信元オブジェクトOb02に着目し（ステップS108）、名前「MainP」に基づいて送信元オブジェクトOb02
10 を定義するシステムクラスC01に着目する（ステップS109）。次に、システムクラスC01に繋がるリソース関連線を探す（ステップS110）。リソース関連線が存在しないので（ステップS111）、元のシステムクラスC01のタイプ<<Interface>>および属性をシステムクラスC01のタイプおよび属性としてそのまま保存する（ステップS113）。

続いて、変換処理部2は、メッセージM02の送信先オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する（ステップS104）。詳しくは、変換処理部2は、まず、送信先オブジェクトOb03に着目し（ステップS108）、名前「Main」に基づいて送信先オブジェクトOb03を定義するシステムクラスC02に着目する（ステップS109）。次に、システム
20 クラスC02に繋がるリソース関連線を探す（ステップS110）。リソース関連線L01が存在するので（ステップS111）、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプとして保存する（ステップS112）。

次に、変換処理部2は、送信元オブジェクトOb02を定義するシステム
25 クラスC01のタイプ<<Interface>>および送信先オブジェクトOb03を定義するシステムクラスC02のタイプ<<Processing>>に基づいて、変換規則格納部4から変換規則d.を検索する（ステップS105）。

続いて、変換処理部2は、変換規則d.「タイプ<<Processing>>

のクラスが定義するリソースノードを配置する。」を適用して、図7に示すように、タイプ<<Processing>>のリソースクラスC11が定義するリソースノードN01に変換して配置する（ステップS106）。

- 次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し
5 （ステップS107）、次のメッセージM03があるので、ステップS102に制御を戻す。

続いて、変換処理部2は、次のメッセージM03をカレントメッセージにセットして着目する（ステップS102）。

- 次に、変換処理部2は、メッセージM03の送信元オブジェクトOb03を
10 定義するシステムクラスのタイプおよび属性を調査する（ステップS103）。

- 詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し（ステップS108）、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する（ステップS109）。次に、システムクラスC02に繋がるリソース関連線を探す（ステップS110）。リソース
15 関連線L01が存在するので（ステップS111）、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプおよび属性として保存する（ステップS112）。

- 続いて、変換処理部2は、メッセージM03の送信先オブジェクトOb04を定義するシステムクラスのタイプおよび属性を調査する（ステップS104）。
20 詳しくは、変換処理部2は、まず、送信先オブジェクトOb04に着目し（ステップS108）、名前「A」に基づいて送信先オブジェクトOb04を定義するシステムクラスC03に着目する（ステップS109）。次に、システムクラスC03に繋がるリソース関連線を探す（ステップS110）。リソース関連線L02が存在するので（ステップS111）、リソース関連のあるリソースクラスC12のタイプ<<Storage>>および属性「評価対象外」をシステムクラスC03のタイプおよび属性として保存する（ステップS112）。
25

次に、変換処理部2は、送信元オブジェクトOb03を定義するシステムクラスC02のタイプ<<Processing>>および送信先オブジェクトOb

04を定義するシステムクラスC03のタイプ<<Storage>>に基づいて、変換規則格納部4から変換規則e.を検索する(ステップS105)。

続いて、変換処理部2は、変換規則e.「タイプ<<Storage>>のクラスが定義するリソースノードを配置する。」を適用して、図7に示すように、タイプ<<Storage>>のクラスC12が定義するリソースノード群N04, N05, N06に変換し、リソースクラスC12の属性が「評価対象」となっているので、ノード群N04, N05, N06を評価対象サブモデル60に配置する(ステップS106)。

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し(ステップS107)、次のメッセージM04があるが、破線矢印のメッセージであるので、これを無視し、さらに次のメッセージの存在を調査し、メッセージM05があるので、ステップS102に制御を戻す。変換処理部2は、次のメッセージM05をカレントメッセージにセットして着目する(ステップS102)。

次に、変換処理部2は、カレントメッセージM05の送信元オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する(ステップS103)。

詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し(ステップS108)、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する(ステップS109)。次に、システムクラスC02に繋がるリソース関連線を探す(ステップS110)。リソース関連線L01が存在するので(ステップS111)、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプおよび属性として保存する(ステップS112)。

続いて、変換処理部2は、メッセージM05の送信先オブジェクトOb05を定義するシステムクラスのタイプおよび属性を調査する(ステップS104)。詳しくは、変換処理部2は、まず、送信先オブジェクトOb05に着目し(ステップS108)、名前「B」に基づいて送信先オブジェクトOb05を定義す

るシステムクラスC04に着目する(ステップS109)。次に、システムクラスC04に繋がるリソース関連線を探す(ステップS110)。リソース関連線L03が存在するので(ステップS111)、リソースクラスC13のタイプ<<Storage>>および属性「評価対象外」をシステムクラスC04のタイプ

5 および属性として保存する(ステップS112)。

次に、変換処理部2は、送信元オブジェクトOb03を定義するシステムクラスC02のタイプ<<Processing>>および送信先オブジェクトOb05を定義するシステムクラスC04のタイプ<<Storage>>に基づいて、変換規則格納部4から変換規則e.を検索する(ステップS105)。

10 続いて、変換処理部2は、変換規則e.を適用しようとするが、システムクラスC04の属性が「評価対象外」となっているので、ノードへの変換を行わずに性能評価サブモデル60に配置しない(ステップS106)。

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し(ステップS107)、次のメッセージM06があるので、ステップS102に

15 制御を戻す。

次に、変換処理部2は、シーケンス図51のカレントメッセージM06に着目する(ステップS102)。

続いて、変換処理部2は、メッセージM06の送信元オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する(ステップS103)。

20 詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し(ステップS108)、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する(ステップS109)。次に、システムクラスC02に繋がるリソース関連線を探す(ステップS110)。リソース関連線L01が存在するので(ステップS111)、リソース関連のあるリソース

25 クラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプおよび属性として保存する(ステップS112)。

次に、変換処理部2は、メッセージM06の送信先オブジェクトOb06を定義するシステムクラスのタイプおよび属性を調査する(ステップS104)。

詳しくは、変換処理部2は、まず、送信先オブジェクトOb06に着目し(ステップS108)、名前「subP」に基づいて送信先オブジェクトOb06を定義するシステムクラスC05に着目する(ステップS109)。次に、システムクラスC05に繋がるリソース関連線を探す(ステップS110)。リソース
5 関連線が存在しないので(ステップS111)、元のシステムクラスC05のタイプ<<Interface>>および属性をシステムクラスC05のタイプおよび属性として保存する(ステップS113)。

次に、変換処理部2は、送信元オブジェクトOb03を定義するシステムクラスC03のタイプ<<Processing>>および送信先オブジェクトOb
10 06を定義するシステムクラスC05のタイプ<<Interface>>に基づいて、変換規則格納部4から変換規則f.を検索する(ステップS105)。

続いて、変換処理部2は、変換規則f.を適用して、外部パッケージのタイプ<<Interface>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価
15 サブモデル60のノードN07に変換し配置する(ステップS106)。このノードN07は、タイプ<<Interface>>のシステムクラスC05が属するパッケージ50を表している。

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し、次のメッセージM07があるが、破線矢印のメッセージであるので、これを無
20 視し、さらに次のメッセージの存在を調査し、メッセージがないと判断する(ステップS107)。

最後に、変換処理部2は、性能評価サブモデル60がシンクノードまたはリターンノードで完結していないので、変換規則h.により、性能評価サブモデル60に終了処理を加える。この性能評価サブモデル60では、最初のオブ
25 ジェクトOb01をソースノードN02としたので、シンクノードN08を配置し、性能評価サブモデル60を完結させる。

なお、第1の実施例では、変換規則a.～h.が変換規則格納部4に格納されているものとしたが、変換規則は変換規則編集部3により変換規則格納部4

に追加および変更可能となっているので、上記に示した変換規則以外に従った変換も可能である。

また、第1の実施例では、1つのパッケージ50の性能評価サブモデル60について説明したが、この処理を繰り返すことにより複数の性能評価サブモデル60からなる性能評価モデル6の全体が同様にして得られる。

第1の実施例では、通常のシステム分析およびシステム設計で行われるような機能的な観点からUMLモデル5を作成し、ハードウェアリソースを表すリソースクラスと、システムクラスとリソースクラスとの関連を示すリソース関連線とを追加することで、性能評価モデル6を生成することが可能となる。これにより、UMLモデル5の作成と性能評価モデル6の作成との距離が縮まり、最終的に、性能評価モデル6の作成時間および作成コストを軽減することが可能となる。

第2の実施例

図8を参照すると、本発明の第2の実施例に係る性能評価モデル生成装置は、図1に示した第1の実施例に係る性能評価モデル生成装置に対して、性能評価モデル生成プログラムを記録した記録媒体100を備える点のみが異なっている。この記録媒体100は、磁気ディスク、半導体メモリ、その他の記録媒体であってよい。

性能評価モデル生成プログラムは、記録媒体100からコンピュータでなる性能評価モデル生成装置に読み込まれ、UMLモデル解析部1、変換処理部2、変換規則編集部3、および変換規則格納部4として動作する。各部1～4の動作は、第1の実施例に係る性能評価モデル生成装置における対応する各部1～4と全く同様になるので、その詳しい説明は省略する。

特許請求の範囲

1. 変換規則を格納する変換規則格納部と、
UMLモデルを入力して解析するUMLモデル解析部と、
前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された
変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有する性能評価モデル生成装置。
2. 利用者からの変換規則を入力する変換規則編集部と、
前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、
UMLモデルを入力して解析するUMLモデル解析部と、
前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された
変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有する性能評価モデル生成装置。
3. UMLモデルの拡張記法を使い、ハードウェアリソースに割り付ける表
記を定め、その表記に従ったUMLモデルから性能評価モデルへの変換規則を
決め、この変換規則によってUMLモデルから系統的に性能評価モデルを作成
する性能評価モデル生成方法。
4. シーケンス図の最初のメッセージをカレントメッセージにセットする工
程と、
シーケンス図のカレントメッセージに着目する工程と、
送信先オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに
基づいて変換規則を検索する工程と、

検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、

カレントメッセージの次のメッセージがあるかどうかを判定する工程と、

次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメッセージに着目する工程に戻す工程と

を含む性能評価モデル生成方法。

5. 前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなる請求項4記載の性能評価モデル生成方法。

6. コンピュータを、変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

7. コンピュータを、利用者からの変換規則を入力する変換規則編集部、前記変換規則編集部により入力された変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

要 約 書

本発明の性能評価モデル生成装置は、変換規則を格納する変換規則格納部と、UMLモデルを入力して解析するUMLモデル解析部と、前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有する。